Generating Better Queries for Systematic Reviews

Harrisen Scells
Queensland University of Technology
Brisbane, Australia
harrisen.scells@hdr.qut.edu.au

Guido Zuccon
Queensland University of Technology
Brisbane, Australia
g.zuccon@qut.edu.au

ABSTRACT

Systematic reviews form the cornerstone of evidence based medicine, aiming to answer complex medical questions based on all evidence currently available. Key to the effectiveness of a systematic review is an (often large) Boolean query used to search large publication repositories. These Boolean queries are carefully crafted by researchers and information specialists, and often reviewed by a panel of experts. However, little is known about the effectiveness of the Boolean queries at the time of formulation.

In this paper we investigate whether a better Boolean query than that defined in the protocol of a systematic review, can be created, and we develop methods for the transformation of a given Boolean query into a more effective one. Our approach involves defining possible transformations of Boolean queries and their clauses. It also involves casting the problem of identifying a transformed query that is better than the original into: (i) a classification problem; and (ii) a learning to rank problem. Empirical experiments are conducted on a real set of systematic reviews. Analysis of results shows that query transformations that are better than the original queries do exist, and that our approaches are able to select more effective queries from the set of possible transformed queries so as to maximise different target effectiveness measures.

KEYWORDS

Systematic Reviews, Query Formulation, Boolean Queries, Query Transformations

ACM Reference Format:
Harrisen Scells and Guido Zuccon. 2018. Generating Better Queries for Systematic Reviews. In *SIGIR '18: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, July 8–12, 2018, Ann Arbor, MI, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3209978.3210020

1 INTRODUCTION

A systematic review is a type of literature review that appraises and synthesises the work of primary research studies (called citations below) to answer one or more research questions. Systematic reviews play a key role in evidence based medicine, informing practice and policy. For example, in order for medical specialists to diagnose and recommend treatment for a patient accurately, they rely on the most up-to-date clinical evidence; similarly systematic reviews are used by government agencies to decide upon health management and policies [15]. Systematic reviews provide this evidence through a comprehensive literature review.

However, the compilation of systematic reviews can take significant time and resources, hampering their effectiveness. With an increasing number of medical studies submitted to databases such as PubMed, it is becoming ever more difficult and laborious for systematic review authors to retrieve relevant studies for inclusion in the review, while minimising the amount of non-relevant citations they need to assess or appraise. Tsafnat et al. report that it can take several years to complete and publish a systematic review [26]. When systematic reviews take such significant time to complete, they can be out-of-date even at the time of publishing.

Retrieval of literature from a medical database for systematic reviews is performed using complex Boolean queries. Boolean queries (and retrieval) are the accepted standard for searching citations when compiling a systematic review [8]. An example of one such query is visible in Figure 1. These queries contain several Boolean operators such as `OR`, `AND`, and `NOT`, as well as advanced operators such as `ADJ` (which matches search terms within a certain range of each other), field restrictions, nesting, and the explosion of terms in an ontology (MeSH). The query in Figure 1 is composed of 19 clauses: each is represented as a line in the query (to which a number is assigned) and may contain one or more operators.

Query formulation is an important step in the definition of the search strategy of a systematic review[1]. A poorly formulated query may retrieve only a subset of the relevant citations to the review study or, conversely, may retrieve an extremely large number of citations, while there may only be few relevant citations. In particular, the retrieval of a large number of citations is often a problem for the compilation of systematic reviews because all of the retrieved citations need to be screened for inclusion in the systematic review (akin to performing relevance assessment). This appraisal phase, commonly performed by two reviewers, is expensive and time consuming, often requiring several person-months to complete [10], thus adding to the costs (both monetary and in terms of time) required for the compilation of a systematic review. Previous work has in fact reported that it can take experienced reviewers between 30 seconds and several minutes [27] to screen a single study (title, abstract and metadata). The effect this has on the timeliness of reviews is highlighted by some notable examples; for example in Shemilt et al.'s *scoping* review, 1.8 million studies were screened, of which only about 4,000 were found to be potentially eligible [24].

SIGIR '18, July 8–12, 2018, Ann Arbor, MI, USA

ACM ISBN 978-1-4503-5657-2/18/07…$15.00
https://doi.org/10.1145/3209978.3210020

[1] Along with a Boolean query, a search strategy also includes which databases are queried, the date the search was performed, and the number of citations retrieved. Note that a search strategy may include more than one Boolean query.

To ensure the queries used by systematic reviews are of high quality[2], reviews often receive the support of information specialists to assist in the query formulation process, and queries are often submitted to an expert panel for review (along with the protocol of the systematic review). However, when formulating queries, little is known about their retrieval performance when applied to answer the questions posed in systematic reviews [1]. Past research has found that only 30% of citations are retrieved using the Boolean queries defined in the protocol of the systematic review [6] (51% of citations were discovered by pursuing references of references, and 24% by personal knowledge or contacts[3]) – a recall problem. On the other hand, past research has also shown that queries may retrieve an overly large set of citations compared to those that were relevant, as it was for Shemilt et al.'s study where only 0.22% of the retrieved citations were relevant [24] – a precision problem.

In this paper, we question whether the Boolean queries used in systematic reviews are the most effective possible (highest recall/-precision), or whether more effective queries are possible and how these can be obtained. Specifically, we seek to answer the following research questions:

RQ1: Is it possible to formulate Boolean queries that are more effective than those originally used within search strategies of systematic reviews? We investigate this with respect to recall, precision, F_β ($F_{0.5}$, F_1, F_3), and work saved over sampling (WSS) as target effectiveness measures.

RQ2: If the answer to RQ1 is positive, then: Can alternative, more effective Boolean queries, generated from the original systematic review queries, be automatically selected?

To answer RQ1, we devise a set of transformations that can be applied to clauses of Boolean queries for generating alternative valid queries to be issued for retrieval. Transformations are applied at the level of clauses and consist in changing the operators used in the original queries. In particular, no new terms are added to the original queries, nor original terms are removed[4]. Through the empirical evaluation of alternative queries generated using the transformations, we show that better Boolean queries are possible, thus answering RQ1 positively.

To answer RQ2 we cast the problem of formulating Boolean queries that are more effective than the original, into two machine learning problems: (1) predicting whether a Boolean query, generated from the original query by applying a chain of query transformations, is more effective than the original query, and (2) ranking the Boolean queries generated from the original query, so that the queries that are better than the original are ranked at the top of the suggested alternative queries. The two problems are tackled by training classification and learning to rank algorithms, respectively.

We empirically show that effective classifiers and rankers can be built, and these can be tailored to optimise different evaluation measures. Specifically, we find that the classifiers identify queries that on average outperform the original queries by 147.72% in precision, 185.13% in $F_{0.5}$, 99.29% in F_1, and 40.45% in F_3; while the rankers identify queries that on average outperform the original queries by 358.47% in precision, 247.79% in $F_{0.5}$, 149.91% in F_1, and 42.90% in F_3. However, neither the classification approach nor the learning to rank approach were able to outperform the original queries (baseline) for recall and work saved over sampling (WSS).

```
1. Diabetic Ketoacidosis/
2. Diabetic Coma/
3. ((hyperglyc?emic or diabet*).tw adj emergenc*.tw.)
4. (diabet*.tw. and (keto* or acidos* or coma).tw.)
5. DKA.tw.
6. or/1-5
7. Insulin Lispro/
8. Insulin Aspart/
9. Insulin, Short-Acting/
10. (glulisine or apidra).tw.
11. (humulin or novolin).tw.
12. (lispro or aspart).tw.
13. (novolog or novorapid).tw.
14. (insulin* adj3 analogue*).tw.
15. acting insulin*.tw.
16. or/7-15
17. 6 and 16
18. (humans/ not exp animals/)
19. 17 and 18
```

Figure 1: A typical Boolean query found in a systematic review. Note that the line numbers form part of query: the query is nested by referring to the line numbers, for instance `or/1-5` on line 6 means that lines 1 through 5 should be combined with a Boolean `OR` operator. The fields to search on (`.tw.`), the Medical subject headings (`/`) and their explosion (subsumption [29] – `exp`) are also encoded in the query.

2 RELATED WORK

Computational methods to automate the process of compiling systematic reviews have recently garnered much attention [4, 11, 17, 18]. While the bulk of prior research has focused on automating the appraisal phase, e.g., the application of active learning for automatically assessing a portion of the retrieved citations, recent attempts have considered the search phase and the application of information retrieval techniques, through the creation of test collections and resources [9, 23] and the improvements of retrieval models and algorithms [17, 22].

With regards to improving queries formulated within systematic review search strategies, two prior works are relevant. Scells et al. [22] exploited field restrictions by limiting search terms to four types of clinical information. The results of this study indicate that this kind of transformation can significantly reduce the total number of citations retrieved by a systematic review. Another attempt at transforming Boolean queries of systematic reviews by Karimi et al. [10] found that simplifying field restrictions and operators, and removing lines from queries offered a significant improvement to recall while slightly degrading precision. They also showed that best-match queries, rather than Boolean, may further increase search effectiveness. One limitation with that study, however, is the authors manually modified the queries and re-ran them. Our work diverges from these two studies, as we seek to automatically apply

[2] i.e. meet standards for effectiveness and bias prior to screening citations for inclusion in the review.

[3] Percentages add up to more than 100% because the same citations appeared in multiple sources.

[4] With the exception of varying the use (i.e. adding or removing) of the MeSH explosion operator. This operator includes in the query all the MeSH terms that are children of the MeSH term to which the operator is applied.

the most effective transformations to Boolean queries, and we are only interested in Boolean queries (i.e. no best-match queries) as these are a widely accepted (and often enforced) standard by the systematic review community.

Techniques such as query generation [12] and query expansion and reduction [16, 25] have seen success in the medical information retrieval space for both clinical and consumer purposes. This work focuses on the clinical space, and does not attempt to modify the structural syntax of a query. Instead, due to the difficulty of formulating complex queries to satisfy the information need of systematic reviews, this work focuses on modifying aspects of a query. For example, it is not known to the researchers formulating a query if a search term should be restricted to the title, the abstract, or both before carrying out the search. The size and complexity of Boolean queries needed to answer the research question outlined in the systematic review is often too great for a research group; especially when the question is difficult. Human error is also a contributing factor in the query formulation process which sometimes introduces error [5, 19, 28] which can lead to less effective queries. A (semi)-automatic query formulation process reduces the risk of introducing errors (e.g. recursive rules, spelling mistakes, or invalid syntax) into the query.

Another challenge once queries have been generated or transformed is selecting the most effective variation. Scells et al. [20] attempted to apply query performance predictors (QPPs) to identify the most effective query variation for best-match systematic review queries. They found that common QPPs are insufficient for identifying effective queries. The application of existing QPPs to the structure of Boolean queries is left for future work.

Of relevance to our work is also the research by Kumaran and Carvalho [14], and Balasubramanian et al. [2] who explored automatic query reduction methods for ad-hoc and web queries. In doing so, they faced similar issues in terms of the feasibility of exploring the full space of query reductions (transformations in our case). To address this, they devised a number of heuristics to limit the search space. In addition, they also explored methods that can automatically select reduced queries, given the original queries, so as to increase search effectiveness.

Finally, Kim et. al. [13] explored Boolean query suggestions in professional search domains, e.g., patent retrieval. They generated Boolean queries from text using decision trees. They then ranked these queries with a Ranking SVM, exploiting query quality predictors as features. This research is notable as it demonstrated the ability for effective Boolean queries to be generated and ranked.

3 METHODS

Next we describe our general method for generating query transformations from the original Boolean query. We then detail the set of considered transformations (Section 3.1), the approach used for query candidate generation (Section 3.3) and those for query candidate selection (Section 3.4), including the features used to represent query candidates for classification and learning to rank.

Figure 2 provides a schematic view of our method to generate alternative Boolean queries. The first step consists of parsing the original Boolean query to extract each clause from the query (remember, a clause is a line of the larger Boolean query, see Figure 1).

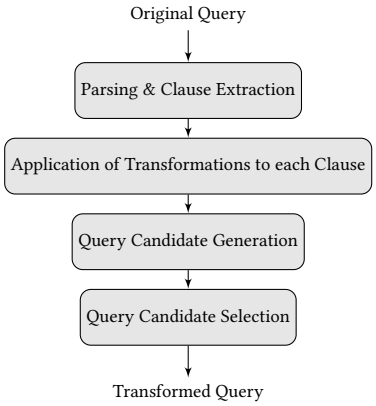

Figure 2: Pipeline of the method used for generating transformations of original systematic review queries.

Note that a clause may contain a reference to one or more previous clauses, as it is the case for line 19 in Figure 1, which combines the results of the clauses at lines 17 and 18 using the operator `AND`.

Once clauses are extracted, one or more transformations are applied to each individual clause, and the resulting transformed clauses recorded, along with the original clause. For example, a transformation that changes `AND` into `OR` may be applied to the clause in line 19 of Figure 1. Note that multiple transformations may be applied. For example, the clause in line 14 could be transformed by replacing or modifying the `ADJ` operator, and by removing the field restriction (`.tw`) – these will count as two transformations being applied.

Original and transformed clauses are then assembled together to form a new candidate query (query candidate generation). For each clause, only one among all clause variations (original and transformed versions of the clause) is selected to form a query; no clause is dropped from or added to a candidate query. This results in each candidate query having the same number of clauses as the original query. The output of this process is a set of candidate queries, which includes the original query.

The next step in the pipeline of Figure 2 is the selection of a candidate transformed query (or the original query) to replace the original query. This is achieved via a candidate selection function. A number of candidate selection functions are described in Section 3.4, including: a ground-truth informed greedy approach, an oracle approach, and the approaches based on classification and learning to rank algorithms. Note that while the first two approaches always select one transformed query, given an original Boolean query, the classification and learning to rank approaches may select more than one transformed query[5] to suggest to the user as better queries.

3.1 Transformations

In order to rewrite a query, we define a set of transformations ($\mathcal{T}$) that are possible for all queries. A transformation is a query rewriting strategy that belongs to one of three classes of transformations: *Query Reduction* (i.e. removing elements from the query), *Query Replacement* (i.e. replacing an element of a query with something else),

[5]The learning to rank approach may be restricted to select only the top ranked query (i.e. top-k = 1).

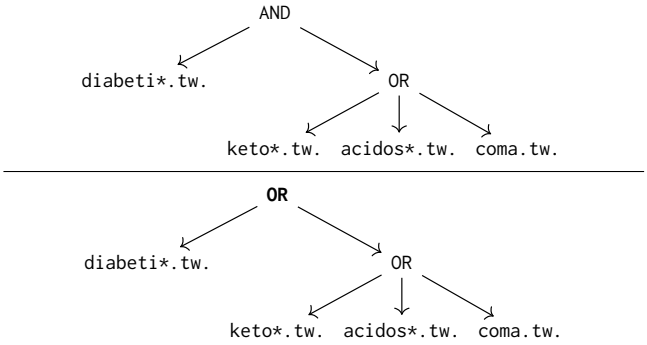

Figure 3: A Logical Operator Replacement transformation applied to a clause. This diagram represents the clause (diabet*.tw. and (keto* or acidos* or coma).tw.) in tree form. Above: the original clause; below: the transformed clause. The replacement type is AND→OR and the operator depth is 0.

and *Query Expansion* (i.e. adding a new element to a query). The desired effect of these transformations is to increase or the reduce the number of citations retrieved, as each type of transformation affects the result set size in different ways. We apply transformations at the query clause level. We also do not consider typical query reduction and expansion strategies (e.g. [25, 29]); however some of the considered transformations are effectively expansions or reductions, e.g., the explosion of MeSH terms.

Next, we describe the family of transformations that we consider in this work (more transformations are possible, but their development and investigation are left for future work). A *family* groups similar transformations together: individual transformations involve specific terms or operators, but the transformations could be represented by the same set of features (features are described in Section 3.5).

3.1.1 Logical Operator Replacement. A query replacement transformation which replaces a single Boolean operator in a query. This transformation considers only the **AND** and **OR** operators[6]. Thus, an **AND** may be replaced by **OR** and vice versa. This transformation is visualised in Figure 3, where the operator depth represents the depth in the logic tree of the Boolean operator that is replaced (depth of root is zero). The transformation **AND→OR** will have the likely impact of increasing recall, often to the expense of precision. The transformation **OR→AND** is likely to have the opposite effect.

3.1.2 Adjacency Range. A query replacement transformation which increases or decreases the range for adjacency operators. Adjacency operators are used in systematic review queries to limit the distance between two or more terms. For example, the clause `(cancer adj2 patient)` requires the term `cancer` to appear at maximum two words away from the term `patient`. This transformation is applied at increasing or decreasing intervals of one. The increase of the adjacency range value is likely to increase recall, while possibly decrease precision. The decrease of the adjacency range value is likely to obtain the opposite effect.

[6]We leave the exploration of the `NOT` operator to future work.

3.1.3 MeSH Explosion. A query expansion and reduction transformation that adds or removes MeSH automatic explosion. MeSH is the Medical Subject Heading ontology, and is used in systematic review queries to retrieve citations that contain the same headings. An "exploded" MeSH heading matches the specified heading, and all of the subheadings in the ontology tree structure, effectively implementing logical subsumption retrieval [29]. The inclusion of the MeSH explosion has the likely effect of increasing recall, at the expenses of increasing the number of citations retrieved (and thus reducing precision). The removal of MeSH explosion is likely to obtain the opposite effect.

3.1.4 Field Restrictions. A query expansion and reduction transformation that modifies the field restrictions for a term in a query clause. Field restrictions are used in systematic review queries to limit term matching to specific fields. This transformation considers the *title* and *abstract* fields (the only two searchable fields that are not metadata). Terms may be restricted in one of three ways: restricting to title only, restricting to abstract only, and restricting to both title and abstract. The use of field restriction has the intent of increasing precision, often at the likely expenses of recall.

3.1.5 Adjacency Replacement. A query replacement transformation which replaces adjacency operators (`ADJ`) with `AND` operators. The intuition for this transformation is that the researcher may have incorrectly used adjacency or incorrectly set the adjacency value to a very restrictive setting, when instead it would have been better to only force the presence of the two terms within a citation rather than within a small window of terms. The Adjacency Replacement transformation has the likely effect of increasing recall, at the possible expenses of precision.

3.2 Application of Transformations

Clause variations are generated on the basis that a transformation can be applied to a clause. Each transformation (e.g., the Logical Operator Replacement) may produce 0 or more candidate clauses. Formally, the set of transformations that can be applied to a clause, denoted as $\mathcal{T}_c'$, is defined as:

$$\mathcal{T}_c' = \forall \tau \in \mathcal{T} | a(\tau, c) = 1 \tag{1}$$

The applicability function a is a Boolean function that determines if an individual transformation τ can be applied to the clause c:

$$a(\tau) = \begin{cases} 1, & \text{if } \tau \text{ is applicable to clause } c \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

The applicability of τ to a clause is determined by aspects of the clause, i.e. a transformation cannot be made if the clause does not contain criteria for the transformation to be applied. Foer example, if the clause does not contain any `ADJ` operators, then the Adjacency Replacement transformation cannot be applied. Clause variations $c_{\tau_1}, c_{\tau_2}, c_{\tau_3} \ldots c_{\tau_m}$, denoted C', are the application of the transformations in $\mathcal{T}_c'$ to the original clause c.

3.3 Query Candidate Generation

Candidate query generation is the process of assembling clauses together to form a Boolean query. For each original clause in the Boolean query, the original clause or one of its variations is selected

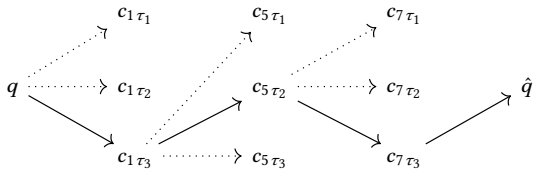

Figure 4: Query transformation chain with three transformations (τ_1, τ_2, and τ_3) being applied to three clauses (c_i) of a query (q) to produce the final rewritten query ($\hat{q}$). Lines between transformations represent possible transformations. Solid lines indicate the path followed (possible transformations for paths that have not been taken are not shown).

to form a candidate query. Figure 4 exemplifies the transformation of an initial Boolean query q into the transformed candidate query $\hat{q}$ through the selection of transformation τ_3 applied to clause c_1, transformation τ_2 applied to clause c_5, and the application of transformation τ_3 again, but applied to clause c_7.

The result of the candidate query generation process for an original Boolean query q is a set of queries $\hat{\mathbb{Q}}_q$ that includes the query q and all possible query transformations. The size of the set of candidate queries depends on the number of clauses and the number of transformations that can be generated for each clause. Assuming that a query q contains n clauses and for each clause a total of maximum m variations (including the original) can be created, then the total number of candidate queries that can be generated for the original query q is $|\hat{\mathbb{Q}}_q| \leq m^n$. Note that this is an upper limit on the number of variations; often in fact different clauses may allow for a different number of variations. While numerous transformations could be applied to some clauses, other clauses may only allow a restricted set of transformations[7]. Because of the high number of possible candidate queries generated, in our empirical evaluation we shall employ a greedy approach to reduce the search space of query candidates. This greedy approach is also used for query candidate selection and is described next among other query candidate selection approaches (Section 3.4.1).

3.4 Query Candidate Selection

Query Candidate Selection is the process by which a query candidate is selected as replacement of the original Boolean query q for a systematic review. A query q^* is chosen from the set of candidate queries $\hat{\mathbb{Q}}_q$ by maximising a *candidate selection function*:

$$q^* = \underset{\hat{q} \in \hat{\mathbb{Q}}_q}{\text{argmax}}\, f(\hat{q}) \tag{3}$$

The candidate selection function $f(\hat{q})$ computes a score for the candidate query $\hat{q}$. In the case of the greedy and oracle approaches of Sections 3.4.1 and 3.4.2, scores are computed by accessing retrieval and ground truth information. In the case of the classification and learning to rank approaches of Sections 3.4.3 and 3.4.4, instead, scores are computed in function of the features used to represent the candidate queries (features detailed in Section 3.5).

While the the greedy and oracle approaches are not applicable in real settings (because they require relevance assessments that would not be available at the time of compiling a new systematic review), they are useful for answering research question RQ1 and provide an indication of upper bound effectiveness attainable[8]. In addition, these two approaches may find applicability in real settings in case a systematic review has been performed in the past, and the search phase needs to be run again, for example to update the results of the systematic review[9]. In these cases, the assessments obtained at the time of compiling the original systematic review may be used as ground truth information for the greedy and oracle approaches.

3.4.1 Greedy Candidate Selection. This approach consists of iteratively selecting query transformations such that at each transformation an objective function is maximised. This is equivalent to traversing the chain of transformations from the original query to a target transformed query. At each iteration i, a subset of candidate queries $\hat{\mathbb{Q}}_q^i \subset \hat{\mathbb{Q}}_q$ is formed by considering all transformations of the previously selected candidate query $\hat{q}^{i-1}$ (the original Boolean query at the first iteration ($i = 1$) of the algorithm, i.e. $\hat{q}^0 = q$) where a transformed query $\hat{q}^i$ differs from $\hat{q}^{i-1}$ for only one clause. Note that, at each iteration (point in the query transformation chain), all possible transformations are applied and run.

In order to select the candidate query at each iteration, the greedy candidate selection function $f(\hat{q})$ is set to compute the number of citations retrieved and select the query that returns the *minimum* number of citations (in the hope of increasing precision). In case of ties, then ground truth information (relevance assessments) is accessed to select the query which maximises the number of relevant citations retrieved. In case of further ties, the query among the tied ones is selected at random.

The greedy candidate selection approach iterates for t times (i.e. t sequences of transformations are applied) or until there is no improvement to the candidate selection function $f(\hat{q})$, whichever condition is met sooner.

During the iterations of the greedy candidate selection approach, subsets $\hat{\mathbb{Q}}_q^0$, $\hat{\mathbb{Q}}_q^1$, ..., $\hat{\mathbb{Q}}_q^t \subset \hat{\mathbb{Q}}_q$ are created. These subsets consist of a portion of the full space of possible query transformations. In our empirical experiments (Section 4) we considered this subset of query transformations as the query candidates from which other approaches, including the oracle, were instructed to select the final query candidate. While this does not represent the entirety of the possible queries, it did provide a convenient way to reduce the search space, making experimentation and evaluation viable. Note that the query candidate selection methods below are independent of the subset of query candidates given as input, and could be applied to subsets obtained in different ways, or, theoretically, to the full set of transformations. A discussion of the implication of this choice is provided in Section 7.

3.4.2 Oracle Candidate Selection. In order to determine an upper bound on the effectiveness of the query transformations, we devise an oracle approach that selects query candidates based on ground truth information (relevance assessments). In particular, we set $f(\hat{q}) = E(\hat{q})$, where $E(\hat{q})$ represents the score assigned to the set of citations retrieved by candidate query $\hat{q}$ by evaluation measure

[7] The exact number of query candidates can be computed as $\prod_{i=1}^{n} m_i$, where m_i is the exact number of variations that can be produced for a specific clause i.

[8] Note that in our empirical experiment this shall not represent the actual maximum effectiveness attainable, as only the subset of query transformations generated using the greedy approach has been explored using the oracle approach, due to the large number of possible permutations of query transformations.

[9] The update of systematic reviews is common.

E, e.g. E may be precision or recall (see Section 4 for a complete list of evaluation measures used – we implemented an oracle for each evaluation measure). Thus, the oracle candidate selection function selects the query $q^* \in \hat{\mathbb{Q}}_q$ that retrieves the set of citations that maximises evaluation measure E.

Next, we describe predictive approaches for the selection of transformed queries from the set of candidate queries, namely a classification-based approach and a learning to rank approach.

3.4.3 Classification-based Candidate Selection. The problem of selecting a transformed query that improves the search results over those from the original Boolean query is cast into the classification problem of predicting whether a transformed query outperforms the original query. By doing so, we implicitly encode a relationship between the transformed query and the original query: whether the former is more effective than the latter. To this aim, we consider pairs of queries formed by pairing the original query q with each of the considered query transformations $\hat{q} \in \hat{\mathbb{Q}}_q$. Then, we aim to predict whether $E(\hat{q}) - E(q) \geq 0$.

To solve this classification problem we devised a Support Vector Machine (SVM) classifier with radial basis function (RBF) kernel $K(\hat{q}, q) = \exp(-\gamma||\underline{\hat{q}}_j - \underline{q}_j||^2)$, where $\underline{\hat{q}}_j$ and $\underline{q}_j$ are values from the feature vectors representing a candidate query $\hat{q}$ and the original query q (features described in Section 3.5). The output of this classifier is a *set of queries* (rather than *a single query* as in the other approaches) that have been predicted to outperform the original Boolean query.

Note that a similar problem definition could be used where, instead of predicting if a query candidate $\hat{q}$ satisfies $E(\hat{q}) - E(q) \geq 0$, a single query candidate q^* (with $q^* \in \hat{\mathbb{Q}}_q$) is selected such that it provides the largest effectiveness gain over the original query q, i.e. $q^* = \text{argmax}_{\hat{q} \in \hat{\mathbb{Q}}_q}[E(\hat{q}) - E(q)]$. This could have been achieved for example with a regressor; we leave the exploration of this approach to future work.

3.4.4 Learning to Rank Candidate Selection. The problem of selecting a transformed query that improves the search results over those from the original Boolean query is cast into a learning to rank problem. In this solution, the objective is to rank query candidates in order to select the most effective query (or top k queries, if a broader setting is considered – this is left for future work).

The learning to rank framework is used to learn a ranking model by training on pairwise preferences between queries, using the target effectiveness measures E. When the pairwise preferences for all candidate queries are collected, a partial ordering among query candidates can be established and the top candidate selected.

To solve this learning to rank problem, we use the Ranking SVM method, where queries are represented using the features described in Section 3.5 and the error function is set to:

$$\phi(\hat{q}_i, \hat{q}_j) = \begin{cases} 1, & \text{if } sign(h(\hat{q}_i) - h(\hat{q}_j)) \neq sign(E(\hat{q}_j) - E(\hat{q}_i)) \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

where the function h is the ranking function to be learnt such that it minimises the overall number of ranking errors:

$$h = \text{argmin} \sum_{q \in \mathbb{Q}} \sum_{\hat{\mathbb{Q}}_q} \phi(\hat{q}_i, \hat{q}_j) \quad (5)$$

where $\mathbb{Q}$ is the set of original queries used to train the ranker and $\hat{\mathbb{Q}}_q$ is the set of query transformations for the original query q (and including q).

3.5 Features

Next, we describe the features used within the classification and learning to rank approaches to represent query candidates. Our approach consists of devising features related to the possible transformations that can be performed on a query. The following are the features that stem from the transformations described in Section 3.1.

Logical Operator Replacement.

Replacement Type: The type of replacement: either AND→OR or OR→AND.

Operator Depth: The depth at which the logical operator appears in the Boolean query.

Adjacency Range.

Distance Change: The change in distance (+1/−1) for the operator. There is no maximum distance, however the minimum absolute value of the distance is 1.

Operator Depth: The depth at which the adjacency operator appears in the Boolean query.

MeSH Explosion.

MeSH Depth: The depth at which the MeSH term appears in the MeSH ontology tree.

Term Depth: The depth at which the MeSH term appears in the Boolean query.

Field Restrictions.

Field Restrictions Type: The type of restriction. There are three restrictions possible (title only, abstract only, both title and abstract) and three starting states: six types of replacements in total.

Field Restrictions Depth: The depth at which the restriction appears in the Boolean query.

Adjacency Replacement:

Operator Depth: The depth at which the adjacency operator appears in the Boolean query.

For a query with multiple transformations, features are computed by inheriting the features of the parent (i.e. the previous query in the chain). If the same transformation is applied multiple times (i.e. it has the same features, with different values), only the most recently applied features are saved. In the extreme case where either (i) no transformations can be applied to the query; or (ii) the resulting candidate queries generated from the transformations do not improve over the original query, the value of the computed features are 0.

4 EXPERIMENTAL SETUP

Collection. We examine one medical literature database (PubMed, one of the most comprehensive databases) and how literature is retrieved using one query language (Ovid, one of the most popular interfaces to PubMed). The experiments performed in this work used a collection of 51 topics[10] from the test collection provided by Scells et al. [23]. Each topic contains a Boolean query extracted

[10]The collection originally contained 94 topics; of these 51 only were selected because these were expressed using the Ovid MEDLINE query language.

from the search strategies of Cochrane systematic reviews that have been deemed high quality.

The CLEF2017 Technology Assisted Reviews (TAR) collection [9] was not used in this work, as it does not contain enough queries for the same search system (i.e. there are not enough Ovid queries – the extension of our query parsers to other query formats is left for future work). However, we use approximately the same number of topics (51 in this work, 50 in total in the TAR collection). Another difference between the collection of Scells et al. (used here) and the TAR collection is that the former considers a wide range of systematic review types, while TAR only considers Diagnostic Test Accuracy (DTA) reviews. Note that like TAR, Scells et al.'s collection also includes as relevant citations that were not retrieved by the query published in the systematic review; these relevant citations were instead sourced by reviewers through personal knowledge or by pursuing references of citations. This means that transformed queries in our experiments may have returned a higher recall value than the original query.

Conducting query transformation experiments for a proprietary query language (Ovid) and database (PubMed) requires the replication of both. A query parser was written for the query language, and at this point any transformation was made. The parsed query was then "compiled" into the Elasticsearch specific query language. The parsing, transformation, and compilation processes were performed automatically, using the experimental framework described in work by Scells et al. [21]. Elasticsearch 5.3.0 was used to index PubMed[11] and its default Boolean search functionalities used.

Evaluation Measures. To evaluate the original systematic review queries and those obtained with the query transformations and selection methods of this paper, we used precision, recall and F_β-measure ($\beta = 1$), for which information retrieval researchers are familiar with. These measures are commonly used to evaluate the effectiveness of the search phase of systematic reviews. We further considered F-measure variants with $\beta = 0.5$ ($F_{0.5}$), which assigns to recall half of the weight than that for precision, and with $\beta = 3$ (F_3), which assigns to recall three times the weight than that for precision. Finally we also used the work saved over sampling measure (WSS) which considers the work saved (with respect to the number of studies required to be screened) by comparing the number of not relevant studies that have not been retrieved (true negatives), those that have been retrieved, and recall [4]. These measures are also commonly used to evaluate the search phase of systematic reviews. Note that we did not use rank-based measures such as average precision (AP) because systematic review users would normally assess the entire set of retrieved citations and would ignore any rank information. Indeed, the underlying retrieval model was the Boolean model, thus not producing a ranked list as output.

Query Generation. The generation of all possible candidate queries for the transformations considered here would have been computationally unfeasible. To maintain the empirical evaluation feasible, we used the greedy candidate selection approach (Section 3.4.1) to create a subset of transformed query candidates, to which we applied the other candidate selection methods. For the greedy candidate selection approach, we set the maximum number of iterations to $t = 5$. The resulting number of total query candidates produced by the greedy candidate selection is 12,686 (average of 248.7 transformed query candidates for each original Boolean query).

Query Candidate Selection. Along with the greedy candidate selection approach, the methods described in Section 3.4 were implemented and evaluated. For the oracle approach, we implemented an oracle method for each evaluation measure considered in this study, resulting in a set of 6 oracle runs.

The library LIBSVM [3] was used to implement the classification-based candidate selection approach. The parameters of the classifier and the RBF kernel (c and γ) were tuned as to maximise the evaluation measures considered in this study. This resulted in a classifier for each target evaluation measure. The availability of a classifier specifically tuned for each evaluation measure would allow for systematic reviewers to select their target evaluation measure according to their use case (e.g., a scoping or rapid review may be well inclined to trading recall for higher precision values). Classifiers were learnt using leave-one-out cross validation, i.e. trained on 50 topics and tested on 1, and repeating this process 51 times until testing has occurred on each topic. Parameters were tuned by performing n-fold cross validation ($n = 5$) on the training data.

The library SVM^{rank} [7] was used to implement the learning to rank candidate selection approach. A pairwise Ranking SVM method was trained and tuned in a similar fashion to the classification-based candidate selection approach, including the creation of a ranker for each target evaluation measure.

5 RESULTS

5.1 RQ1: Are Better Queries Possible?

To answer RQ1, we explored the effectiveness of query transformations generated from the original query and compared their effectiveness. Given the exponential amount of transformed queries that were possible to generate, we used the greedy approach to both generate candidate query transformations and to select candidates; candidate selection was also performed using the oracle approach on the generated queries. The results are reported in Table 1, where ^{b} refers to statistical significant differences between the retrieval effectiveness of the queries obtained with the considered method and the original Boolean query (*Baseline*). Statistical significance was computed using a two-tailed t-test and $p < 0.01$.

In answer to our first research question, we found that it was possible to generate Boolean queries via the transformations illustrated in this work that provided higher effectiveness than the original Boolean queries used in the considered high-quality systematic reviews. Specifically, the greedy approach selected transformed queries that on average improved the baseline's precision and all F-measure variants (though with no statistical significant differences). However, these queries traded off recall and resulted in a lower WSS value. The oracle approaches, instead, consistently outperformed the baseline, with most improvements being statistically significant. This highlights how significant the increase of target effectiveness could be if the right query is selected. In addition, note that the oracle may not have selected the globally optimal transformed queries, as only the subsets of queries generated during the greedy candidate selection process were considered.

[11] Downloaded on 1st January 2017 as per specifications in Scells et al.'s collection [23].

	Precision	Recall	$F0.5$	$F1$	$F3$	WSS
Baseline	0.010	0.815	0.012	0.018	0.053	0.801
Greedy	0.034 (+236.33%)	0.725 (-11.11%)	0.034 (+184.24%)	0.039 (+114.74%)	0.068 (+27.29%)	0.720 (-10.10%)
Oracle_{p}	**0.078 (+680.68%)**^{b}	0.521 (-36.06%)^{b}	0.077 (+537.16%)^{b}	0.086 (+380.70%)^{b}	0.146 (+174.71%)^{b}	0.520 (-35.10%)^{b}
Oracle_{r}	0.011 (+8.98%)	**0.938 (+15.04%)**^{b}	0.012 (+0.86%)	0.015 (-15.35%)	0.027 (-49.97%)	0.865 (+7.99%)
$\text{Oracle}_{F0.5}$	0.076 (+663.67%)^{b}	0.543 (-33.37%)^{b}	**0.078 (+540.90%)**^{b}	0.088 (+389.45%)^{b}	0.153 (+188.80%)^{b}	0.542 (-32.37%)^{b}
Oracle_{F1}	0.073 (+633.48%)^{b}	0.545 (-33.13%)^{b}	0.077 (+534.48%)^{b}	**0.089 (+394.48%)**^{b}	0.155 (+192.26%)^{b}	0.544 (-32.12%)^{b}
Oracle_{F3}	0.062 (+522.25%)^{b}	0.577 (-29.21%)^{b}	0.068 (+463.08%)^{b}	0.084 (+367.42%)^{b}	**0.159 (+199.48%)**^{b}	0.576 (-28.15%)^{b}
Oracle_{WSS}	0.023 (+133.45%)	0.925 (+13.41%)	0.024 (+99.43%)	0.027 (+48.91%)	0.047 (-12.10%)	**0.903 (+12.69%)**
Classifier_{p}	0.025 (+147.72%)^{o}	0.693 (-15.02%)	0.028 (+129.36%)	0.035 (+95.90%)	0.075 (+40.18%)^{b}	0.681 (-15.01%)
Classifier_{r}	*0.035 (+253.09%)*	*0.717 (-12.11%)*^{o}	*0.037 (+203.61%)*	*0.042 (+132.98%)*	0.073 (+38.25%)	*0.711 (-11.22%)*
$\text{Classifier}_{F0.5}$	0.032 (+224.53%)	0.713 (-12.57%)	0.035 (+185.13%)^{o}	0.040 (+125.70%)	0.077 (+44.45%)	0.704 (-12.11%)
Classifier_{F1}	0.026 (+157.78%)	0.693 (-14.98%)	0.029 (+136.51%)	0.036 (+99.29%)^{o}	0.075 (+40.21%)^{b}	0.682 (-14.85%)
Classifier_{F3}	0.026 (+160.37%)	0.692 (-15.11%)	0.029 (+138.90%)	0.036 (+101.19%)	0.075 (+40.45%)$^{b\,o}$	0.681 (-15.00%)
Classifier_{WSS}	0.033 (+229.41%)	0.684 (-16.11%)	0.035 (+188.95%)	0.041 (+129.61%)	*0.080 (+50.85%)*^{b}	0.671 (-16.22%)^{o}
Ranker_{p}	0.046 (+358.47%)	0.653 (-19.90%)^{b}	0.043 (+252.78%)	0.044 (+147.11%)	0.065 (+22.97%)	0.649 (-18.97%)^{b}
Ranker_{r}	0.034 (+237.78%)	*0.753 (-7.70%)*^{o}	0.033 (+172.19%)	0.036 (+101.95%)	0.058 (+9.60%)	*0.748 (-6.64%)*
$\text{Ranker}_{F0.5}$	0.046 (+362.90%)	0.619 (-24.12%)^{b}	0.042 (+247.79%)	0.043 (+141.94%)	0.062 (+16.95%)	0.615 (-23.24%)^{b}
Ranker_{F1}	0.048 (+380.17%)	0.614 (-24.69%)^{b}	0.043 (+257.07%)	0.045 (+149.91%)	0.067 (+26.71%)	0.610 (-23.83%)^{b}
Ranker_{F3}	*0.056 (+455.82%)*	0.645 (-20.92%)^{b}	*0.051 (+319.32%)*	0.053 (+197.77%)	0.076 (+42.90%)	0.641 (-20.00%)^{b}
Ranker_{WSS}	0.053 (+428.60%)	0.610 (-25.25%)^{b}	0.050 (+314.01%)	*0.054 (+199.11%)*	*0.081 (+52.34%)*	0.607 (-24.27%)$^{b\,o}$

Table 1: Effectiveness of Baseline (original query) and transformed queries selected using Greedy, Oracle, Classifier, and Ranker. The measure optimised for (the Oracle, Classifier, and Ranker) is denoted with $_{p}$ (precision), $_{r}$ (recall), $_{F_{0.5}}$, $_{F_1}$, $_{F_3}$ (F_β measures), and $_{WSS}$ (WSS). Values in bold denote the best value obtained for that evaluation measure; values in italics the best values for each type of approach. ^{b,o} indicate statistical significant differences compared to Baseline (b) and Oracle (o).

5.2 RQ2: Can Better Queries be Automatically Selected?

To answer RQ2, we explored the effectiveness of a classification (*Classifier*) and a learning to rank approach (*Ranker*). The results are reported in Table 1 (last two row-groups), where ^{o} refers to statistical significant differences (two-tailed t-test, $p < 0.01$) between the retrieval effectiveness of the considered method and the query obtained by the Oracle (we only compare methods trained for an effectiveness measure E to the Oracle obtained for the same effectiveness measure E). (^{b} is as for Section 5.1). Note that while the Ranker approach only selected one query candidate, the Classifier approach may have selected more than one: the reported effectiveness is the average effectiveness obtained by all candidate queries selected by the Classifier. We study the variance in effectiveness among different queries in the selected set in the next section.

In answer to our second research question, we found that the devised automatic methods outperformed the original Boolean query (Baseline) for each of the target measure they were tuned for, except for recall and WSS. For these measures, in fact, no gains were recorded and for Rankers most of the losses were statistically significant. The statistical analysis also highlighted that:

- while gains were made, there were no statistical significant differences between the effectiveness of the automatic methods and the Baseline (except for Recall and WSS);
- yet, percentage improvements over the Baseline for precision and F-measure variants were consistent;
- while the automatic methods could not reach the effectiveness of the Oracle, most differences were not statistically significant;
- some settings of the automatic methods outperformed the Greedy approach (though not significantly).

The results and considerations above highlight that while queries that improve over the original recall values are possible, as shown by the Oracle_{r} results, the transformed queries obtained with the automatic methods could not select queries that improved recall, even when tuned for this job.

6 FURTHER ANALYSIS AND DISCUSSION

Next, we further analyse the results with respect to: (i) the average number of transformed clauses applied to the original queries to generate the selected candidate queries, (ii) the selection of original queries by the query candidate selection approaches, (iii) the variation in effectiveness among candidate queries selected by the classifier approach, (iv) the position assigned by the rankers to the query selected by the Oracle approach (the 'best' query in our experiments).

6.1 Number of Transformed Clauses

The candidate query selection methods select transformed versions of the original queries. The query generation method used in the experiments generated queries with up to $t = 5$ transformed clauses. Table 2 reports the average number of transformed clauses that were counted in the candidate queries selected by each query selection approach. Considering the oracle, the number of transformed clauses for the identified best query varied according to the target measure: recall required the transformation of the least number of clauses, while WSS and F_3 of the most. The same trend is found when analysing the learning to rank approach. However, this is not the case when considering the queries selected by the classification

Method	Avg. Tr. Cl.	Method	Avg. Tr. Cl.
Baseline	0	Greedy	2.4
Oracle_p	3.3	Oracle_{F1}	3.3
Oracle_r	2.3	Oracle_{F3}	3.4
$\text{Oracle}_{F0.5}$	3.3	Oracle_{WSS}	3.4
Classifier_p	2.3	Ranker_p	3.3
Classifier_r	2.6	Ranker_r	2.2
$\text{Classifier}_{F0.5}$	2.6	$\text{Ranker}_{F0.5}$	3.3
Classifier_{F1}	2.4	Ranker_{F1}	3.3
Classifier_{F3}	2.3	Ranker_{F3}	3.4
Classifier_{WSS}	2.6	Ranker_{WSS}	3.5

Table 2: Average number of transformed clauses that have been used for the candidate queries selected by each method. (Baseline is 0 because it refers to the original queries).

approach, for which the least transformed clauses were found when considering precision and F_3, and the most when considering all other measures but F_1. In general, the number of transformations found in queries selected by the classification method were lower than those for the oracle and the learning to rank.

6.2 Selection of Original Queries

In our experiments, the set of candidate queries from which the selection occurs also contained the original Boolean query. We thus investigated how many times the original query was selected by each candidate selection approach. Out of 51 topics, the Greedy approach selected the original queries 3 times. The Oracle approach selected the original query once when tuned for precision and all F-measure variants, and twice when tuned for recall and WSS. Out of all the queries selected by the Classifier, none were the original one. The Ranker approach only selected the original query for 1 of the 51 topics when tuned for recall and F_1.

6.3 Classification for Query Selection

Unlike the other methods, the classification approach for query selection returned a set of candidate queries. The distribution of number of queries selected per topic for each classifier is shown in Figure 5. On average, we found the classification approach returned 122.62 queries per topic. Of these, 49.51% were incorrectly selected, i.e. they were not more effective than the baseline (for the target measure the Classifier was tuned for). Figures 6 and 7 report the distribution of true positive (selected queries that are more effective than the original) and false positive for each classifier. We observe that the classifiers tuned for F_1 and precision were the most error-prone: 51.3% and 51.1% of selected queries were false positives, respectively. Conversely, we found that the classifier tuned for recall had the highest classification performance: only 14% of all selected queries were false positives (did not improve over the baseline). Yet, those false positives had the most detriment in terms of average retrieval effectiveness: the queries selected by the Classifier had a recall lower than that of the original query.

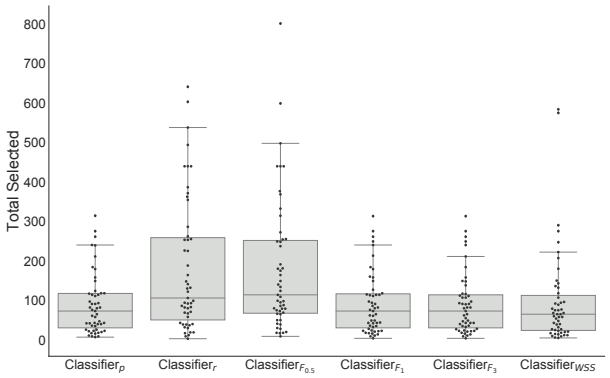

Figure 5: Distribution of the number of candidate queries selected by the classifiers for the topics in the collection.

6.4 Learning to Rank for Query Selection

The learning to rank approach ordered candidate queries by decreasing predicted effectiveness, and selected the top-ranked query. We further analysed the query rankings obtained by this method to identify the average rank position in which the best query was placed (i.e. the query that actually returned the highest effectiveness for the target measure, and that was selected by the Oracle approach). Figure 8 shows the distribution of the rank at which the best query was placed by each of the six rankers. The figure suggests that for most topics, the 'best' queries is ranked among the few top queries for the ranker tuned for WSS; while for other rankers the 'best' query is often ranked far off the top. This analysis suggests that further improvements may be obtained, at least for WSS, if the learning to rank selection approach considered $k > 1$ top queries, rather than $k = 1$ queries.

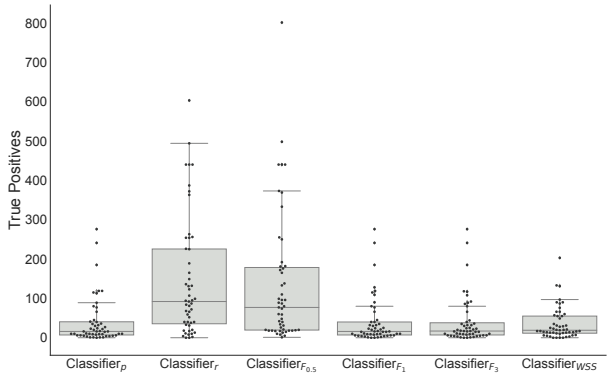

Figure 6: Distribution of true positive classifications for each classifier.

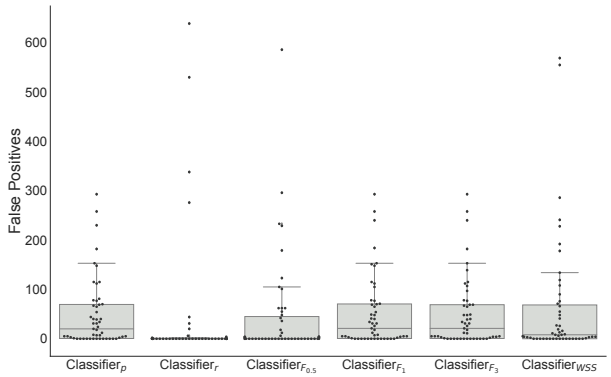

Figure 7: Distribution of false positive classifications for each classifier.

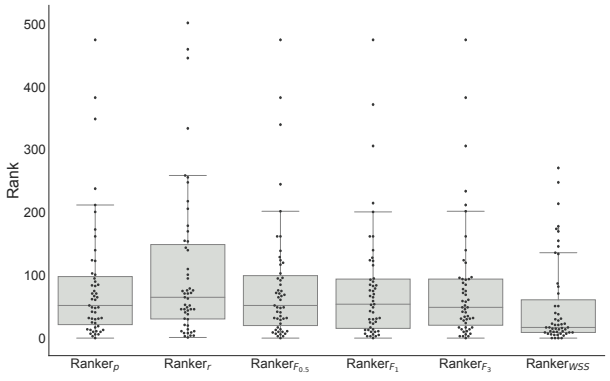

Figure 8: Rank distribution of the 'best' query within the query ranking obtained with each Ranker.

7 CONCLUSIONS

In this paper we considered Boolean queries used within systematic reviews and explored whether there exist better Boolean queries given a target effectiveness measure to be used within the search phase. To this aim, we developed automatic query transformation methods alongside query selection methods aimed at automatically identify queries that would outperform the original one.

Through empirical evaluation using a collection of queries and assessments extracted from 51 high-quality systematic reviews, we found that better Boolean queries are possible, and our methods for automatically identifying these better queries improve the search and appraisal phases of existing systematic reviews. Our method of transforming Boolean queries via a query transformation chain fills the gap that other research has performed manually [10].

While we were able to generate better queries than the original using classification and learning to rank approaches, some measures (predominately recall), were found to be difficult to optimise for. We hypothesise that, with more features and training examples, the effectiveness of these models may improve. Nevertheless, we have shown that it is possible to optimise for other measures, and these optimised models do outperform the original query.

An open problem is whether the subset of query candidates generated by the greedy selection approach is representative enough of the queries attainable using other approaches. In addition, to apply the methods of this paper in a real setting, the greedy approach to reduce the search space should be modified by removing the need of the ground truth information (which we use only in presence of ties to further simplify the process). The greedy approach could still be used when considering re-iterating the search phase for updating existing systematic reviews.

For future work we will investigate more transformations that could be applied within the same query transformation chain approach, such as query expansion techniques via, for example, relevance feedback and pseudo relevance feedback. Additionally, we seek to identify more features for existing transformations, in order to improve the training process. Finally, we plan to perform these experiments on a larger set of queries, and on other existing collections, such as the CLEF2017 TAR collection [9].

We envision the methods developed by the line of research initiated in this paper to be integrated as a query assistance tool to assist systematic reviewers at query formulation time.

Acknowledgements. The authors would like to thank Daniel Locke, Anton van der Vegt and Jimmy for insightful comments on early drafts of this work. Harrisen is the recipient of a CSIRO PhD Top Up Scholarship. Dr Guido Zuccon is the recipient of an Australian Research Council DECRA Research Fellowship (DE180101579) and a Google Faculty Award.

REFERENCES

[1] T. Agoritsas, A. Merglen, D.S. Courvoisier, C. Combescure, N. Garin, A. Perrier, and T.V. Perneger. 2012. Sensitivity and predictive value of 15 PubMed search strategies to answer clinical questions rated against full systematic reviews. *JMIR* 14, 3 (2012), e85.
[2] N. Balasubramanian, G. Kumaran, and V.R. Carvalho. 2010. Exploring reductions for long web queries. In *SIGIR'10*. ACM, 571–578.
[3] C.C. Chang and C.J. Lin. 2011. LIBSVM: A library for support vector machines. *TIST* 2 (2011), 27:1–27:27. Issue 3.
[4] A.M. Cohen, W.R. Hersh, K. Peterson, and P.Y. Yen. 2006. Reducing workload in systematic review preparation using automated citation classification. *JAMIA* 13, 2 (2006), 206–219.
[5] S. Golder, Y. Loke, and H. M. McIntosh. 2008. Poor reporting and inadequate searches were apparent in systematic reviews of adverse effects. *Journal of Clinical Epidemiology* (2008).
[6] T. Greenhalgh and R. Peacock. 2005. Effectiveness and efficiency of search methods in systematic reviews of complex evidence: audit of primary sources. *BMJ* 331, 7524 (2005), 1064–1065.
[7] T. Joachims. 2006. Training linear SVMs in linear time. In *SIGKDD'06*. 217–226.
[8] Green S JPT CHH. 2011. Cochrane handbook for systematic reviews of interventions version 5.1.0 [updated March 2011]. *The Cochrane Collaboration* (2011).
[9] E. Kanoulas, D. Li, L. Azzopardi, and R. Spijker. 2017. CLEF 2017 Technologically Assisted Reviews in Empirical Medicine Overview. In *CLEF'17*.
[10] S. Karimi, S. Pohl, F. Scholer, L. Cavedon, and J. Zobel. 2010. Boolean versus ranked querying for biomedical systematic reviews. *BMC MIDM* 10, 1 (2010), 1.
[11] M. Khabsa, A. Elmagarmid, I. Ilyas, H. Hammady, and M. Ouzzani. 2016. Learning to identify relevant studies for systematic reviews using random forest and external information. *ML* 102, 3 (2016), 465–482.
[12] S.N. Kim, D. Martinez, L. Cavedon, and L. Yencken. 2011. Automatic classification of sentences to support evidence based medicine. *BMC bioinformatics* 12, 2 (2011).
[13] Y. Kim, J. Seo, and W. B. Croft. 2011. Automatic boolean query suggestion for professional search. In *SIGIR'11*.
[14] G. Kumaran and V.R. Carvalho. 2009. Reducing long queries using query quality predictors. In *SIGIR'09*. 564–571.
[15] J. Lavis, H. Davies, A. Oxman, J.L. Denis, K. Golden-Biddle, and E. Ferlie. 2005. Towards systematic reviews that inform health care management and policy-making. *JHSRP* 10, 1_suppl (2005), 35–48.
[16] G. Luo, C. Tang, H. Yang, and X. Wei. 2008. MedSearch: a specialized search engine for medical information retrieval. In *CIKM'08*. 143–152.
[17] D. Martinez, S. Karimi, L. Cavedon, and T. Baldwin. 2008. Facilitating biomedical systematic reviews using ranked text retrieval and classification. In *ADCD'08*.
[18] M. Miwa, J. Thomas, A. O'Mara-Eves, and S. Ananiadou. 2014. Reducing systematic review workload through certainty-based screening. *JBI* 51 (2014), 242–253.
[19] M. Sampson and J. McGowan. 2006. Errors in search strategies were identified by type and frequency. *JCE* 59, 10 (2006), 1057–e1.
[20] H. Scells, L. Azzopardi, G. Zuccon, and B. Koopman. 2018. Query Variation Performance Prediction for Systematic Reviews. In *SIGIR'18*.
[21] H. Scells, D. Locke, and G. Zuccon. 2018. An Information Retrieval Experiment Framework for Domain Specific Applications. In *SIGIR'18*.
[22] H. Scells, G. Zuccon, B. Koopman, A. Deacon, L. Azzopardi, and S. Geva. 2017. Integrating the framing of clinical questions via PICO into the retrieval of medical literature for systematic reviews. In *CIKM'17*.
[23] H. Scells, G. Zuccon, B. Koopman, A. Deacon, S. Geva, and L. Azzopardi. 2017. A Test Collection for Evaluating Retrieval of Studies for Inclusion in Systematic Reviews. In *SIGIR'2017*.
[24] I. Shemilt, A. Simon, G.J. Hollands, T.M. Marteau, D. Ogilvie, A. O'Mara-Eves, M.P. Kelly, and J. Thomas. 2014. Pinpointing needles in giant haystacks: use of text mining to reduce impractical screening workload in extremely large scoping reviews. *RSM* 5, 1 (2014), 31–49.
[25] L. Soldaini, A. Cohan, A. Yates, N. Goharian, and O. Frieder. 2015. Retrieving medical literature for clinical decision support. In *ECIR'15*. 538–549.
[26] G. Tsafnat, P. Glasziou, M.K. Choong, A. Dunn, F. Galgani, and E. Coiera. 2014. Systematic review automation technologies. *SR* 3, 1 (2014), 74.
[27] B.C. Wallace, T.A. Trikalinos, J. Lau, C. Brodley, and C.H. Schmid. 2010. Semi-automated screening of biomedical citations for systematic reviews. *BMC Bio* 11, 1 (Jan 2010), 55.
[28] A. Yoshii, D.A. Plaut, K.A. McGraw, M.J. Anderson, and K.E. Wellik. 2009. Analysis of the reporting of search strategies in Cochrane systematic reviews. *JMLA* 97, 1 (2009), 21.
[29] G. Zuccon, B. Koopman, A. Nguyen, D. Vickers, and L. Butt. 2012. Exploiting Medical Hierarchies for Concept-based Information Retrieval. In *ADCS'12*.